06 | 物模型：如何定义智能电灯？

2020-11-20 郭朝斌

物联网开发实战 进入课程 >

讲述：郭朝斌
时长 17:47 大小 16.29M

你好，我是郭朝斌。

在基础篇最后一讲的智能家居项目里，我们设计了几个小场景，其中就包括智能电灯。如果你只是想自娱自乐，做一个可以用手机 App 控制的电灯，那么只要通过代码实现控制功能就足够了。至于是怎么控制的，电灯有什么状态上报，你自己知道就行了。

但是，如果你想让智能电灯真正成为物联网系统的一部分，那就不仅仅是在封闭的、确定的场景下写几行代码的事儿了。在物联网平台上，可能有其他人开发的应用需要显示你的智能电灯的状态；也可能有别的设备，比如光照传感器、智能音箱，在场景联动中要控灯的状态。

所以，你需要把控制电灯打开和关闭的方法，告诉这些应用和产品的开发人员。同时，这些开发人员也需要了解，智能电灯的状态信息如何获取和解析。那么，你面临的第一个问题就是，**用什么方式提供这些接口信息呢？**

另外，市面上不止一款智能电灯，如果要一一适配，那工作量肯定很大，而且扩展起来会很困难。那么，你面临的第二个问题就是，**平台应用如何避免针对每款智能灯进行定制开发呢？**

计算机领域的软件体系结构采用的是一种**层**的结构，所以有人说过这么一句名言：**“计算机科学领域的任何问题，都可以通过增加一个间接的中间层来解决。”**

按照这个思路，我们就可以在智能电灯实体和平台之间，增加一层标准规范来解决这些问题。就像，你使用不同的浏览器访问极客时间的网站，都可以看到课程的文本、音频、视频等内容，因为这些内容都是基于 **HTML** （HyperText Markup Language，超文本标记语言）等规范组织的。

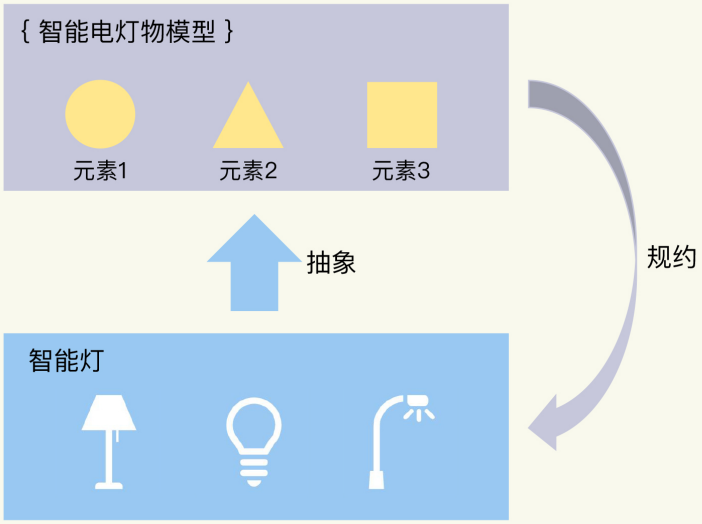
物联网中的这层规范就是 Thing Specification Language，简称 **TSL**。使用 TSL 描述的物联网中的实体模型，就是**“物模型”**，或者叫做“产品模型”，也有叫“数据模板”的。不过，我认为“物模型”更有物联网专属的感觉，所以在咱们这门课里我都会用“物模型”这个叫法。

物模型和设备的关系是什么？

物模型是物理世界的实体东西的一个抽象，是进行数字化描述后，用于数字世界的数字模型。这么说可能有点绕，更直接一点说就是，物模型是使用计算机可以理解的语言，说清楚这个产品**是什么**、**能做什么事情**，以及**可以提供哪些信息**。

而抽象就是要提取出产品的共同特征，形成模型。以智能灯为例，不同的灯，尽管规格不同，但它们的属性是相似，比如都有开关状态的属性，功能逻辑也相仿。我们可以将这些特征标准化，形成智能灯的物模型。

反过来，物模型也规约了设备的功能。新增加的设备，如果是同一类型的，在设计、研发中，会遵循相同的功能定义，有相同的特征，实现相同的服务。比如，灯都应该有“开”和“关”两种状态。



为什么要使用物模型?

基于共同的抽象特征，物模型可以让应用程序不再针对一个个的产品设备，而是同一类设备采用相同的处理逻辑。这实际上是**应用开发的基础**。当烟感传感器的数值触发报警时，即使是不同品牌的烟感产品，应用程序也可以对数值做相同的处理和判断，否则只能分别进行数值分析。

另外，物模型中，设备的功能是明确定义的，可以方便地实现场景联动。比如，光线传感器可以基于光照强度，向智能电灯发送亮度的控制命令，或者开和关的命令。

如何定义物模型?

那么，如何定义智能电灯的物模型呢？这里我想告诉你结论，我们一般是通过属性、事件和动作这三种功能元素来定义。接下来，我就一一和你介绍。

我们知道，智能电灯的状态，要么是打开，要么是关闭；当进行控制时，这两种状态还会相互转换。此外，有些灯还可以根据需求设置不同的亮度、颜色和色温等。

它们的共同点就是，都描述了产品设备运行时的某种状态，我们用**属性（Property）**来表示。

属性的特点是可读可写。也就是说，应用程序可以读取属性，也可以设置设备的属性。我们还可以看到类似的例子，比如环境监测设备的温度、湿度这两个属性等。

如果智能电灯在运行过程中，出现了低电压的情况，或者发生了硬件故障，那么联网的设备可以将这些信息发送出去，通知你来及时作出处理。

这类由产品设备在运行过程中产生的信息、告警和故障等，就是**事件（Event）**。

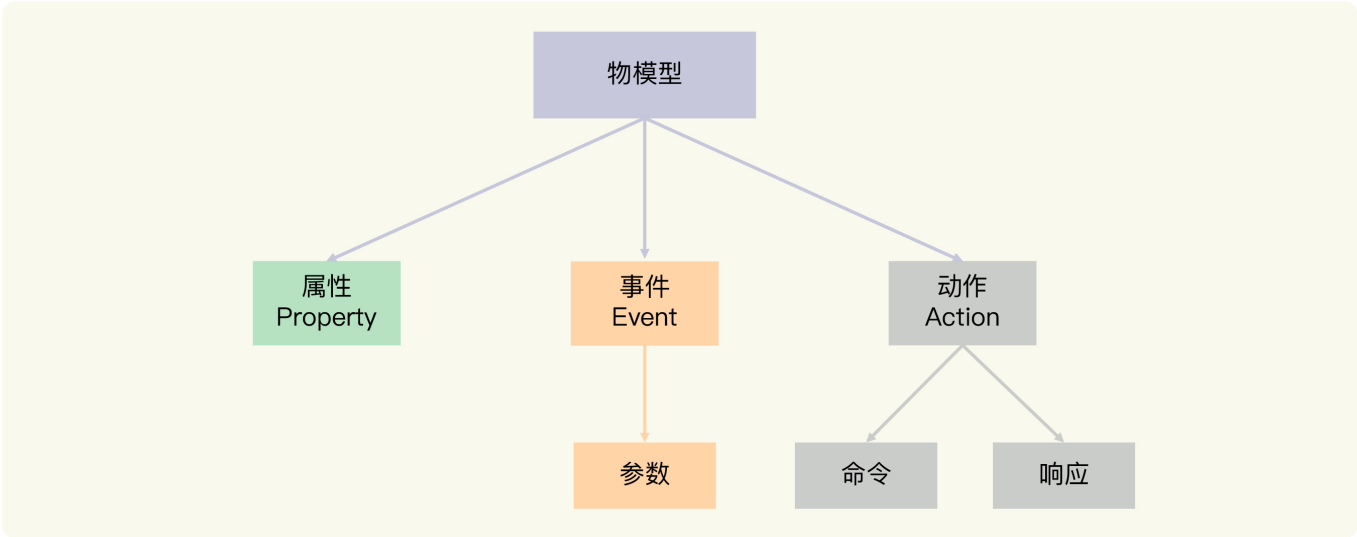
一个事件可以包含多个输出参数。事件不同于属性，事件是设备上报的，不能由应用来设置。类似的例子，还有某任务完成时的消息，环境传感器检测到污染物的告警等。

我们再看生活中关于灯的一个使用场景：第一次约会的时候，你希望灯能够烘托出浪漫的气氛，就要调节灯的颜色、亮度和色温。如果分别设置属性，将会非常繁琐，这时你会想到要为灯增加一个场景模式的功能，一个命令就可以设置到浪漫模式。

这种设备可以被调用的能力或者方法，就是**动作（Action）**，也被称作**服务（Service）**。

动作由应用下发给设备，设备可以返回结果给应用。从执行的流程看，动作还可以进一步分为同步和异步。这取决于动作是否是个耗时的操作，以及其他应用逻辑对于动作执行结果的依赖关系。

你可能想，设置属性也可以改变设备的状态，那它们的区别是什么呢？相比于属性，动作是应用下发到设备的控制命令；动作可通过一条指令实现更复杂的业务逻辑，比如，调低温度 5 度，旋转摄像头 30°等。



到这里，我们定义了属性、事件和动作这三类功能，也就完成了物模型的定义。

接下来，我们要做的是通过数据来描述它们。和编程语言一样，作为一种模型语言，物模型的数据也有不同的数据类型。它们主要包括六种：

1. **布尔型**（Bool）：非真即假的二值型变量。例如，开关功能只有开、关两种状态。
2. **整数型**（Int）：可用于线性调节的整数变量。例如，电灯的亮度是一个整数范围。
3. **字符串型**（String）：以字符串形式表达的功能点。例如，灯的位置。
4. **浮点型**（Float）：精度为浮点型的功能点。例如，电压值的范围是 0.0 - 24.0。
5. **枚举型**（Enum）：自定义的有限集合值。例如，灯的颜色有白色、红色、黄色等。
6. **时间型**（Timestamp）：String 类型的 UTC 时间戳。

对于整数型、浮点型的数值，它们的单位可以是百分比、电压、米等。

物模型一般是用 **JSON 格式**来表述模型元素。JSON 是 Web 开发中，经常使用的数据格式，相比于 XML，它更加简洁、清晰，也更轻量级。

在实践中，你手动写完 JSON 格式的物模型后，可以使用检测工具来验证语法是否正确，比如，在线检测工具 JSON Schema Lint。

接下来，我们就按照属性、事件、动作 / 服务这三个要素，一起看看如何用 JSON 格式来定义智能电灯的物模型吧。

定义智能电灯的物模型

智能电灯的开关属性是布尔类型，是必须有的属性。它可以通过 JSON 表述如下：

复制代码

```
{
      "id": "power_switch",    //属性的唯一标识
      "name": "电灯开关",       //名称
      "desc": "控制电灯开灭",    //属性的详细描述
      "required": true,        //表示此属性是否必需包含，是
      "mode": "rw",            //属性的模式，r代表读，w代表写
      "define": {              //属性的数值定义
        "type": "bool",        //数值的类型，布尔
        "mapping": {           //具体数值的含义
          "0": "关",            //0表示灯关闭
          "1": "开"             //1表示灯打开
```

```
12          }
13        }
14      }
```

智能电灯的电压是需要监控的数值，当电压低时，可以上报这个事件。这个事件有一个参数，即电压值，数据类型是浮点类型。JSON 格式的描述如下：

复制代码

```
{
        "id": "low_voltage",        //事件唯一标识
        "name": "LowVoltage",        //事件名称
        "desc": "Alert for device voltage is low",   //事件的描述
        "type": "alert",            //事件的类型，告警
        "required": false,          //表示此属性是否必需包含，否
        "params": [                  //事件的参数
          {
            "id": "voltage",          //事件参数的唯一标识
            "name": "Voltage",        //事件参数的名称
            "desc": "Current voltage",   //参数的描述
            "define": {                  //参数的数值定义
              "type": "float",           //数值类型，浮点数
              "unit": "V",               //数值的单位，伏
              "step": "1",               //数值变化的步长，1
              "min": "0.0",               //数值的最小值
              "max": "24.0",              //数值的最大值
              "start": "1"                 //事件的起始值
            }
          }
        ]
      }
```

动作的定义，和属性、事件的定义过程类似，这里我就不再单独解释了。我们直接将所有属性、事件和动作合并，就得到了智能电灯物模型的完整 JSON 格式：

复制代码

```
{
  "version": "1.0",              //模型版本
  "properties": [                //属性列表
    {
      "id": "power_switch",      //电灯开关属性
      "name": "电灯开关",
      "desc": "控制电灯开灭",
      "required": true,
      "mode": "rw",
```

```
      "define": {
        "type": "bool",
        "mapping": {
          "0": "关",
          "1": "开"
        }
      }
    },
    {
      "id": "brightness",          //亮度属性
      "name": "亮度",
      "desc": "灯光亮度",
      "mode": "rw",
      "define": {
        "type": "int",
        "unit": "%",
        "step": "1",
        "min": "0",
        "max": "100",
        "start": "1"
      }
    },
    {
      "id": "color",               //电灯颜色属性
      "name": "颜色",
      "desc": "灯光颜色",
      "mode": "rw",
      "define": {
        "type": "enum",
        "mapping": {
          "0": "Red",
          "1": "Green",
          "2": "Blue"
        }
      }
    },
    {
      "id": "color_temp",          //色温属性
      "name": "色温",
      "desc": "灯光冷暖",
      "mode": "rw",
      "define": {
        "type": "int",
        "min": "0",
        "max": "100",
        "start": "0",
        "step": "10",
        "unit": "%"
      }
    }
  ],
  "events": [                          //事件列表
```

```
    {
      "id": "status_report",          //运行状态报告
      "name": "DeviceStatus",
      "desc": "Report the device status",
      "type": "info",
      "required": false,
      "params": [                      //事件参数列表
        {
          "id": "status",
          "name": "running_state",
          "desc": "Report current device running state",
          "define": {
            "type": "bool",
            "mapping": {
              "0": "normal",
              "1": "fault"
            }
          }
        },
        {
          "id": "message",
          "name": "Message",
          "desc": "Some extra message",
          "define": {
            "type": "string",
            "min": "0",
            "max": "64"
          }
        }
      ]
    },
    {
      "id": "low_voltage",            //低电压告警事件
      "name": "LowVoltage",
      "desc": "Alert for device voltage is low",
      "type": "alert",
      "required": false,
      "params": [
        {
          "id": "voltage",
          "name": "Voltage",
          "desc": "Current voltage",
          "define": {
            "type": "float",
            "unit": "V",
            "step": "1",
            "min": "0.0",
            "max": "24.0",
            "start": "1"
          }
        }
      ]
```

```
    },
    {
      "id": "hardware_fault",              //硬件错误事件
      "name": "Hardware_fault",
      "desc": "Report hardware fault",
      "type": "fault",
      "required": false,
      "params": [
        {
          "id": "name",
          "name": "Name",
          "desc": "Name like: memory,tf card, censors ...",
          "define": {
            "type": "string",
            "min": "0",
            "max": "64"
          }
        },
        {
          "id": "error_code",
          "name": "Error_Code",
          "desc": "Error code for fault",
          "define": {
            "type": "int",
            "unit": "",
            "step": "1",
            "min": "0",
            "max": "2000",
            "start": "1"
          }
        }
      ]
    }
  ],
  "actions": [],                    //动作列表
  "profile": {                      //产品参数
    "ProductId": "8D1GQLE4VA",      //产品ID
    "CategoryId": "141"              //产品分类编号
  }

```

每个模型都要从头定义吗？

那我们在创建自己的新模型时，是不是每次都需要从头定义这些属性、事件和动作呢？有没有更简便的方式呢？答案当然是有的。

创建模型的时候，有拷贝和继承两种模式，这两种创建模式的不同主要体现在**模型关系**上。

“拷贝”模式类似于编程语言中的值拷贝，新建模型与被拷贝模型有完全相同的三元素，两个模型相互独立，模型变更互不影响。

“继承”模式就是面向对象编程中的继承概念，新建模型被定义为“子模型”，被继承的模型定义为“父模型”。

继承的具体特征是：

1. 子模型继承父模型的所有要素，且继承的元素无法被修改。
2. 子模型可以再被继承，支持多层的继承关系。
3. 子模型可以创建独立的要素，但子模型中新增的要素不可以和所有上级父模型中的元素重名。
4. 当父模型中的元素发生变更时，子模型中继承自父模型的元素同步变更，保持与父模型一致。

以我们刚刚定义的智能电灯的物模型为例，如果要增加安装位置的属性，可以继承已有的模型，然后再增加安装位置的属性。（注意：下面的 JSON 表述省略了与父模型重复的内容。）

复制代码

```
{
...
{
      "id": "name",                    //灯位置属性
      "name": "灯位置名称",
      "desc": "灯位置名称：书房、客厅等",
      "mode": "rw",
      "required": false,
      "define": {
        "type": "string",
        "min": "0",
        "max": "64"
      }
    }
...
}
```

到这里，我们已经了解了物模型，并且完整实践了一遍物模型的创建。接下来，我给你延伸一下，讲两个和物模型相关的概念。

物模型的拓展应用

你也许听到过“设备影子”和“数字孪生”这两个概念，它们和我们这里说的“物模型”是什么关系呢？

设备影子

设备影子用于**缓存设备状态**。应用程序可以通过设备影子直接获取设备最后一次更新的属性值，而无需每次都访问设备。设备在线时，可以直接获取应用指令；设备离线后，再次上线可以主动拉取应用指令。

我们可以再想象一个场景。如果设备网络稳定，很多应用程序请求获取设备状态，设备需要根据请求响应多次，即使响应的结果是一样的。但是可能设备本身处理能力有限，其实无法负载被请求多次的情况。

使用设备影子机制，设备只需要主动同步状态给设备影子一次，多个应用程序请求设备影子获取设备状态，即可获取设备最新状态，做到应用程序和设备的解耦。

再比如，智能电灯的开关状态这个属性，手机 App 可以远程控制，你也可以在本地通过物理开关改变。如果网络不稳定，那么平台上存储的状态，和电灯设备的真实状态可能会不一致，导致后续操作逻辑错误。

设备影子可以通过双向的同步，实现服务器端和设备端属性的一致，从而解决这个问题。

数字孪生（Digital Twin）

物模型是物理实体的数字化模型，但主要针对的是物联网中应用的开发和设备的互操作。

这个模型如果更进一步，集成了物理实体的各类数据，那就是物理实体的**忠实映射**。同时，在物理实体的整个生命周期中，它会和实体一起进化，积累各种信息和知识，并且促进物理实体的优化。这样的模型就是物理实体的数字孪生。

在工业物联网领域，这个概念已经有了很多的探讨和应用。

比如，特斯拉公司为其生产的每一辆电动汽车都建立了数字孪生模型，相关的模型数据保存在公司的数据库中，以便在测试中排查故障，为用户提供更好的服务。

小结

总结一下，在这一讲中，我通过智能电灯的例子讲解了物模型。

1. 物模型是物理世界中产品设备的数字化模型，它对设备的共同特征进行了抽象，同时规约了设备的设计。
2. 物模型一般是使用 TSL 描述的 JSON 格式文件。
3. 物模型包括属性、事件和动作三个功能元素。其中，属性可读可写；事件可以包括多个参数；动作包括应用下发的命令，和设备返回的响应信息

在实践中，定义物模型时，你需要注意物模型三个功能元素的区别，尤其要了解属性和动作的联系和不同。不好的定义会给功能实现带来困难，比如，将智能电灯的“开”和“关”，定义为两个不同的动作。

物模型在物联网系统开发中，作用重大，它为应用开发提供了统一的数据模板，方便了场景联动的实现，同时，为平台上实现设备影子提供了基础。

类似地，数字孪生也正是建立在物理实体的数字模型之上的重要技术方向。这里作为一个引子，有兴趣的话，你可以深入了解一下，也许对你在工作中做系统设计有帮助。

思考题

最后，我还是给你留个思考题作为结尾。

物模型是实战开发的基础，咱们最后再通过一个练习来强化下学习效果吧。请你定义一个环境温湿度传感器的物模型。你可以从属性、事件、动作三个元素的角度思考一下，而且一定要动手写一写。

欢迎在留言区写出你的答案，和我一起交流一下。在后面的实战中，我们也会涉及到温湿度传感器的物模型。如果你的朋友对物联网有兴趣，也欢迎你将这个课程分享给他们一起学习进步。

提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 05 | 系统实例：怎样设计一个简易物联网系统?

下一篇 07 | 零配置组网：设备如何发现彼此?

精选留言 (13)

写留言

doulefi123
2020-11-20

物模型给我启发太大了，因为我是半路出家，现在我们的设备用mqtt订阅了多条消息（开、关等各种动作），如果改成物模型，订阅一条消息就可以了，相对经济些。再加上场景控制联动，能实现自由形式的功能。

展开

作者回复: 谢谢你的留言，多交流!

5

Socrakit
2020-11-22

//环境温湿度传感器
{
"version": "1.0",
"properties": [
{...

展开

作者回复: ❖❖❖❖❖❖赞
温度、湿度的mode可以改为只读r

2

胡黎红
2020-11-23

tsl是标准么?

展开

作者回复: 还没有统一的标准，各家平台都有自己的定义。

1

贤伟
2020-11-20

Lwm2m 协议中把物理实体都看作资源（resource），可以使用 XML 格式定义资源集合（标准和扩展的 ）。使用路径Object/ObjectInstance/Resource/ResourceInstance 来操作资源（读、写、执行、订阅）。我理解和文中的属性相似，只是已经集成进了协议里，作为M2M沟通的通用语言。

展开

作者回复: ❀❀❀❀❀❀ 是的，LwM2M是基于IPSO定义的这些，相比来说，它要比MQTT和CoAP这些应用层更进一步。

2 1

钦开
2020-11-20

老师，您觉得边缘计算和物联网是一种什么关系呢?

展开

作者回复: 这个在网关那节我会讲到。简单说物联网需要边缘计算。

1 1

莹
2020-12-04

```
{
  "version": "1.0", //模型版本
  "properties": [ //属性列表
```

{...

展开

碎步马拉松

2020-12-02

数字孪生 = 设备影子 + 设备全生命周期数据？

展开

作者回复: 数字孪生是个内涵外延很广的概念，不止这些。它试图在数字世界完全映射实体物理世界的东西。比如大众点评上的一个餐馆，也可以认为是一个真实餐馆的数字孪生，有它的菜品、位置、电话、大家的评价等各种描述信息。如果你的餐馆没有在系统上存在孪生对象，那可以说在数字世界，它不存在。那说回设备，如果一个设备要在数字世界有孪生对象。就会用到很多物联网技术，这个孪生对象可以具体收集的数据分析出很多信息，比如设备的震动幅度最近有增大趋势，采用几种不同技术后，再跟进一段时间数据，可以对比出哪种方式好，从而推进设备的技术迭代，等等很多应用。

cras

2020-11-30

基于物模型的定义，云端实现设备影子。状态同步确实是一个难点，如何保证状态一致性老师能否展下说下。

作者回复: 嗯，有一个中间角色，大家都以它为准，这就是保证一致性的本质。

1

marquis

2020-11-27

```
// 温湿度传感器的物模型设计
{
  "version": "1.0", //模型版本
  "properties": [ //属性列表
    {...
```

展开

作者回复: ❖❖❖❖❖❖大赞

marquis
2020-11-26

谢谢郭老师。在TSL物模型中，我主要学习到三个关键参数“属性，事件和动作”之间的关系。
用自己的话总结：属性，是产品能到达的某种状态。而有了属性，使用者才能通过应用去控制产品到达某种属性状态，这就是动作。事件就是产品在遇到主动完成属性切换或者被动检测到属性变换事件，上报给应用的信息。

展开

作者回复: ❖❖❖❖❖❖

Nior
2020-11-25

结合厂商实际文档加深理解：
华为IoT物模型文档：https://support.huaweicloud.com/usermanual-iothub/iot_01_0017.html
小米IoT物模型文档：https://iot.mi.com/new/doc/standard/miot-spec/spec

展开

作者回复: ❖❖❖❖❖❖

Lee
2020-11-22

老师，能否推荐一些物联网书籍、博客？

展开

编辑回复: 相关资料会请老师专门在之后的加餐中整理补充，请保持对课程的关注☺

185
2020-11-22

新学到：设备影子、数字孪生。多谢。

展开

作者回复: ❖❖❖❖❖❖